# Exercises: ASP.NET Core – Exploring Core Features

Problems for exercises for the ["ASP.NET Core Advanced" course @ SoftUni](https://softuni.bg/trainings/4954/asp-net-advanced-june-2025)

A popcorn and film reels and a movie ticket

Description automatically generated with medium confidence

## Introduction

This document is designed to guide you through the core functionalities of the Cinema Web Application. During the workshop, we will explore the application together, reviewing the existing structure and focusing on each functionality one by one. For every feature, you will find a clear explanation of what it does, how it behaves, and what role it plays in the system. This is not a step-by-step coding tutorial—it is a practical companion that will help you understand and implement each functionality as we discuss and analyze it together in the workshop.

## User Functionalities

In this part of the workshop, we will explore the functionalities available to users of the application — both **guests** (not logged in) and **logged-in users**. We will go through the relevant parts of the application that define what users can do and how they interact with movies, cinemas, tickets, and personal watchlists

### Home

The HomeController is responsible for **displaying the landing page of the application**. This is the first page users see when they access the site. It is available to everyone — **both guests and logged-in users**.

#### Action: Index()

* Provides a visual entry point into the system.
* Displays static or introductory content, such as a welcome message or branding.
* May contain navigation links or buttons that direct users to other sections: Cinemas, Movies, Watchlist (if logged in), etc.
* No data is loaded from the database in this action.



A screenshot of a movie theater

AI-generated content may be incorrect.

### Cinema

The CinemaController is responsible for displaying available cinemas to all users and showing the movie program for each cinema. Both actions are accessible without authentication, meaning they can be used by **guests** as well as **logged-in users**.

#### Action: Index()

* **Access Level:** Public ([AllowAnonymous])
* **URL:** /Cinema/Index or simply /Cinema
* Loads all cinemas using the service layer.
* If no cinemas are found, an empty list is returned.
* The view typically displays basic cinema information:

This page allows users to explore the available cinemas before deciding what to watch.



A screenshot of a computer

AI-generated content may be incorrect.

#### Action: Program(Guid id)

* **Access Level:** Public ([AllowAnonymous])
* **URL:** /Cinema/Details/{id}
* Loads cinema details and its current movie program using the given cinema ID.
* If the cinema or movie list cannot be found, the user is redirected back to the cinema list.

This action allows users to focus on one specific cinema and see what is currently being shown there.



A screenshot of a movie screen

AI-generated content may be incorrect.

#### Action: Details(Guid id)

* **Access Level:** Public ([AllowAnonymous])
* **URL:** /Cinema/Program/{id}
* Receives a cinema ID (Guid) and loads detailed information about that cinema.

This view is intended to give users a complete overview of the selected cinema before they view the program or decide to buy tickets.



A movie poster on a screen

AI-generated content may be incorrect.

### Movie

The MovieController handles functionality related to browsing and viewing movie details. All actions in this controller are accessible to both guests and logged-in users. This controller allows users to explore the full list of movies in the system and view individual movie details.

#### Action: Index()

* **Access Level:** Public ([AllowAnonymous])
* **URL:** /Movie/Index or simply /Movie
* Calls the service to **retrieve all movies**.

This is the main page where users can browse the complete movie catalog.



A screenshot of a computer

AI-generated content may be incorrect.

#### View Movie Details in Modal (Partial View)

* When a user clicks **View Details**, the app loads movie details into a modal.



A movie on a screen

AI-generated content may be incorrect.

### Watchlist

The WatchlistController is responsible for managing the **personal movie watchlist** of each user. This feature is only available to **logged-in users**, and all actions are protected with the [Authorize] attribute.

#### Action: Index()

* **Access Level:** Logged-in users only ([Authorize])
* **URL:** /Watchlist/Index
* Retrieves the current user's ID.
* Calls the service to get a list of all movies the user has added.
* Returns a view that lists those movies

This is the main view where users manage their personal movie favorites.



A screenshot of a computer

AI-generated content may be incorrect.

#### Action: AddToWatchlist(string? movieId)

* **Access Level:** Logged-in users only
* Checks for user ID and valid movie ID.
* Calls the service to perform the add operation.
* If successful: redirects to Index with a success message.
* If unsuccessful: shows error message and redirects to /Movie

A white rectangular object with black lines

AI-generated content may be incorrect.

#### Action: RemoveFromWatchlist(string? movieId)

* **Access Level:** Logged-in users only
* Retrieves user ID and checks the movie ID.
* Calls the service to remove the movie.
* Redirects back to the Index view.

### Ticket

Users can view the tickets they have purchased and buy new ones (if available) for movies in a specific cinema. This functionality is split into **two controllers**:

* TicketController – Displays a logged-in user’s owned tickets
* TicketApiController – Handles ticket purchases via an API call (used behind the scenes)

#### TicketController

##### Action: Index()

* **Access Level:** Logged-in users only ([Authorize])
* **URL:** /Ticket/Index or /Tickets
* Retrieves the logged-in user’s ID.
* Calls the ticket service to fetch all tickets bought by that user.
* Displays each ticket entry



A black and grey rectangular object

AI-generated content may be incorrect.

#### TicketApiController

##### Endpoint: BuyTicket([FromBody] BuyTicketRequest request)

**Access Level:** Logged-in users only ([Authorize])

**HTTP Method:** POST

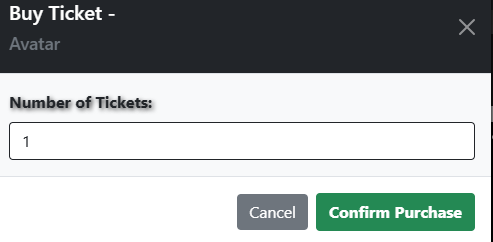
**Route:** /api/TicketApi/BuyTicket

**Purpose:** Allows authenticated users to purchase one or more tickets for a specific movie in a specific cinema.



A screenshot of a movie screen

AI-generated content may be incorrect.



## Manager Functionalities

This section describes the functionalities available to users who have the **Manager role in the system**. Managers are responsible for maintaining the structure and availability of the cinema system, including managing cinemas, configuring movie programs, setting up showtimes, and adjusting ticket availability.

**These functionalities go** **beyond what regular users can access**. Each one is designed to support real-world administrative operations such as scheduling, resource planning, and monitoring.

### Home

The HomeController provides a simple entry point to the **Manager Dashboard**. From here, the manager can access all other configuration tools: cinema programs, movie showtimes, and ticket availability.

#### Action: Index()

* **Access Level:** Managers only ([Authorize(Roles = "Manager")])
* **URL:** /Manager/Home/Index
* Loads a static dashboard view.
* Presents the manager with three main options:
* **Configure Program** – Manage which movies are assigned to each cinema
* **Configure Showtimes** – Adjust the showtimes and screening schedules
* **Configure Tickets** – Set and update the number of available tickets



A screenshot of a computer dashboard

AI-generated content may be incorrect.

### Configure Cinemas

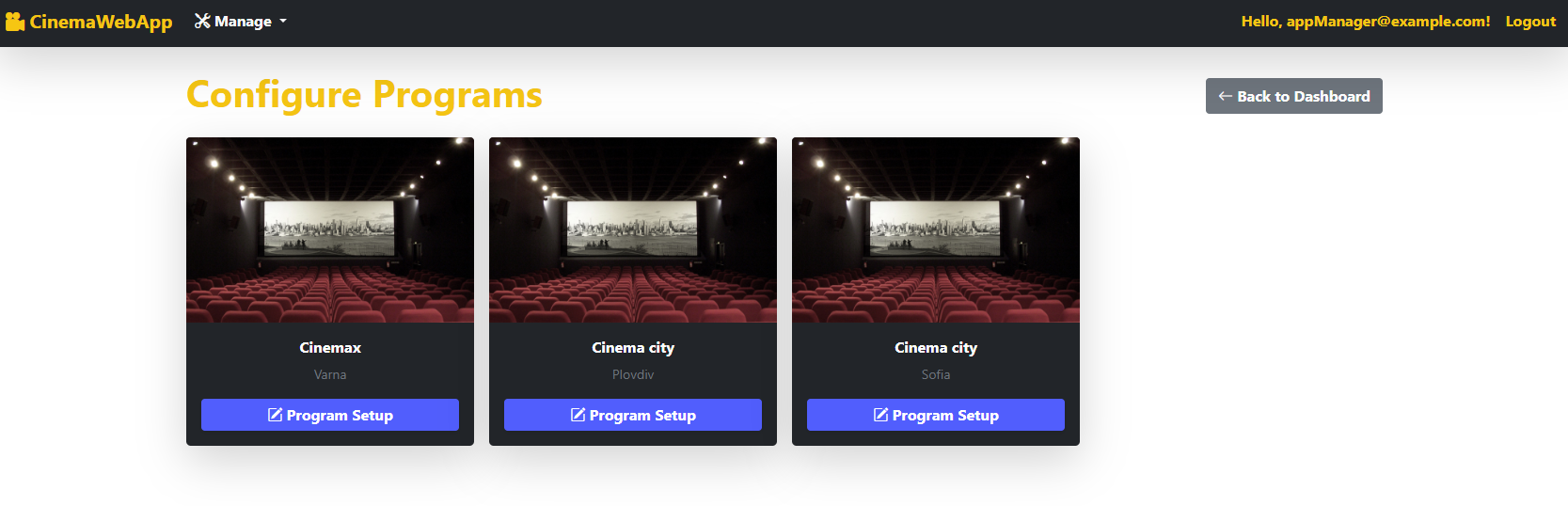
The ConfigureCinemasController allows managers to view and manage all cinemas in the system. This is typically the first step before assigning movies or setting up tickets. Only users with the **Manager** role can access this functionality.

#### Action: Index()

* **Access Level:** Managers only ([Authorize(Roles = "Manager")])
* **URL:** /ConfigureCinemas/Index or /ConfigureCinemas
* • Retrieves all cinemas from the database.
* • Displays each cinema as a card with:
* Cinema name
* Location
* A button labeled Program Setup

This page is the **starting point** for managing each cinema's movie schedule





### Program Setup

The ProgramSetupController allows managers to configure which movies are included in the program for a specific cinema. This feature provides full control over the movie schedule by letting managers add or remove movies from a cinema's offering.

#### Action: Index(Guid cinemaId)

* **Access Level:** Managers only ([Authorize(Roles = "Manager")])
* **URL:** /ProgramSetup/Index?cinemaId={id}
* **Purpose**: Displays a list of all movies with toggle switches to include or exclude them from the program for the selected cinema.

The list is presented as an interactive table. Managers simply flip switches to control which movies are part of the cinema's current program.



A screenshot of a program

AI-generated content may be incorrect.

### Configure Movie Showtimes

Once the cinema program is set up, managers can fine-tune the projection times for each movie in each cinema. This functionality ensures that each cinema offers the right movies at the right times based on its unique schedule.

#### ShowtimeSetupController

##### Action: Index()

* **Access Level:** Manager only
* **URL:** /Manager/ShowtimeSetup
* Retrieves all cinemas that already have movies included in their movie program.
* Sends the list of cinemas to the view as a model.
* Each cinema is displayed in a card with its name, location, and a button labeled Adjust Showtimes.
* The view allows managers to pick a cinema and proceed to configure showtimes for its assigned movies.
* This is the entry point for adjusting showtimes and does not yet involve selecting specific showtimes — it focuses on choosing the cinema first.



A screenshot of a computer

AI-generated content may be incorrect.

#### ShowtimeApiController

##### Endpoint: GetMoviesByCinema(Guid cinemaId)

**• Route:** None (loaded dynamically inside /Manager/ShowtimeSetup)

**• Displayed In:** Index.cshtml using JavaScript (manageShowtimes.js)

**• Purpose:**

* Displays a table of all movies in the selected cinema.
* Shows a list of predefined showtime options (e.g., 12 PM, 3 PM…).
* Allows the manager to check or uncheck available time slots.
* Each movie row includes a Save button, which sends the selected showtimes to the API.
* Uses the endpoint /Manager/api/ShowtimeApi/UpdateShowtimes via AJAX (POST).



**Function: openManageShowtimesModal(cinemaId, cinemaName, cinemaLocation)**

* Makes a GET request to: /Manager/ShowtimeSetup/GetMoviesWithShowtimes
* Loads all movies for a specific cinema.
* Injects the modal HTML into the DOM.
* Opens the modal using Bootstrap’s .modal('show').

A screenshot of a tv show

AI-generated content may be incorrect.

##### Endpoint: UpdateShowtimes([FromBody] ShowtimeViewModel model)

* **Access Level:** Authorized users ([Authorize])
* **HTTP Method: POST**
* **Route:** /Manager/api/ShowtimeApi/UpdateShowtimes
* **Purpose:**
* Accepts the updated list of showtimes for a single movie in a cinema.
* Validates the input model.
* Calls the service to update the showtimes in the database.
* Responds with success or error message.

### Manage Tickets (Manager)

Managers can configure ticket availability (how many tickets can be sold) for each movie in a specific cinema. This functionality is split into two controllers:

* **TicketController** – Displays all cinemas available to the manager, allowing them to begin the ticket setup process.
* **TicketApiController** – Handles ticket availability updates via API (used behind the scenes in modals or JS-driven UI).

#### TicketController

##### Action: Index()

* **Access Level:** Managers only ([Authorize])
* **URL:** /Manager/Ticket/Index or /Manager/Ticket
* Retrieves all cinemas (sorted by location).
* Displays each cinema with a **Manage Tickets** button.



A screenshot of a computer

AI-generated content may be incorrect.

#### TicketApiController

##### Endpoint: GetMoviesByCinema(Guid cinemaId)

* **Access Level:** Managers only ([Authorize(Roles = "Manager")])
* **HTTP Method:** GET
* **Route:** /Manager/api/TicketApi/GetMoviesByCinema/{cinemaId}
* **Purpose:**
* Retrieves all movies currently included in the selected cinema’s program.
* Returns the data as a JSON array.
* Used to populate the **Manage Tickets** modal, listing all movies for that cinema.





A screenshot of a computer

AI-generated content may be incorrect.

##### Endpoint: UpdateAvailableTickets([FromBody] SetAvailableTicketsViewModel model)

* **Access Level:** Managers only ([Authorize(Roles = "Manager")])
* **HTTP Method:** POST
* **Route:** /Manager/api/TicketApi/UpdateAvailableTickets
* **Purpose:**
  + Accepts a request body containing:
    - CinemaId (Guid)
    - MovieId (Guid)
    - AvailableTickets (int)
  + Validates the incoming data using model state.
  + Update ticket capacity.
  + If the operation succeeds, returns 200 OK with a success message.
  + If it fails, returns 400 Bad Request with an error message.

## Admin Functionalities

This section covers the functionalities available to administrators. Admins have full control over the application’s data, including the ability to manage users, cinemas, and movies. Through the Admin Dashboard, they can access each management area to perform actions such as adding new entries, editing existing records, or removing outdated content to keep the system organized and up to date.

### Home (Admin)

The Admin Home Page serves as the central hub for all administrative actions. It features a clean and functional dashboard layout with three main sections: **Manage Users**, **Manage Cinemas**, and **Manage Movies**. Each section is presented as a card with a brief description and a navigation button, allowing the admin to quickly access and manage key parts of the application.

#### Action: Index()



#### A screenshot of a computer AI-generated content may be incorrect.

### Cinema Management

Admins have full control over the cinema data. They can add new cinemas, update existing ones, or soft delete them. All actions are available through a management interface that lists all existing cinemas.

A screen shot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screen shot of a computer screen

AI-generated content may be incorrect.

### Movie Management

Admins can view all movies, create new entries, update existing data, or remove movies from the system. The interface is table-based and includes quick access buttons for editing or deleting each entry.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a movie

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

### User Management

This section allows admins to view all registered users, assign or remove roles (User/Manager), and delete accounts when necessary. Admin accounts are protected and cannot be modified or deleted from this interface.



A screenshot of a computer

AI-generated content may be incorrect.